Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be “used for any purpose other than private study, scholarship, or research.” If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of “fair use” that user may be liable for copyright infringement.

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation.

Printing note: If you do not wish to print this page, then select “Pages from: first page # to: last page #” on the print dialog screen.
The Van Houten library has removed some of the personal information and all signatures from the approval page and biographical sketches of theses and dissertations in order to protect the identity of NJIT graduates and faculty.
ABSTRACT

The specific purpose of this thesis is the automated recognition of the off-line Chinese hand-printed characters by using a blue ball-point pen. Through mask processing, the main components in a Chinese character such as vertical, horizontal, and slant strokes can be extracted. Then, the connected components with the coordinates of the top, bottom, leftmost, and rightmost ends of each stroke extracted are found. From these coordinates, the length and position of each stroke can be computed.

According to the number, relative length, and relative position of each stroke, both of the coarse and fine rule-based classification can be made, and the goal of this thesis is able to be reached.

Excluding the load and segmentation of the original image, the computing time for the feature extraction and classification depends on the image size and the number of strokes. It is about 0.3 seconds per Chinese character on an IBM PC 80486 DX33.

The advantages of the proposed method include efficient time complexity, strong ability to detect very similar Chinese characters, tolerance of the slope of the stroke, and 96% or higher recognition rate.

The disadvantage is the inflexibility for learning driven by the users since the matching rules are open to the manufactures only at present.
OFF-LINE HAND-PRINTED
CHINESE CHARACTER RECOGNITION
BASED ON STROKE MATCHING

by
Mr. Sunshine Chang

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

Department of Computer and Information Science

January 1995
APPROVAL PAGE

OFF-LINE HAND-PRINTED
CHINESE CHARACTER RECOGNITION
BASED ON STROKE MATCHING

Sunshine Chang

Dr. Frank Y. Shih, Thesis Advisor
Associate Professor of Department of
Computer and Information Science

Dr. Peter A. Ng, Committee Member
Professor and Chairman
Department of Computer and Information Science

Dr. James McHugh, Committee Member
Graduate Advisor and Ph.D. Program Director
Department of Computer and Information Science
BIOGRAPHICAL SKETCH

Author: Mr. Sunshine Chang
Degree: Master of Science in Computer Science
Date: January 1995
Date of Birth:
Place of Birth:

Undergraduate and Graduate Education

- Master of Science in Computer Science, New Jersey Institute of Technology, Newark, New Jersey, 1995
- Bachelor of Science in Computer Science, Tamkang University, Taiwan, R.O.C., 1990

Major: Computer Science
This thesis is dedicated to my parents.
ACKNOWLEDGMENT

The author wishes to express his sincere gratitude to his advisor, Dr. Frank Shih, for his guidance, friendship, and moral support throughout this research.

Special thanks to Professors Peter A. Ng and James McHugh for serving as members of the committee.

The author appreciates the timely help and suggestion from his classmates, friends, relatives, and families.

And finally, a thank you to all of the authors and their advisors listed in the references of this thesis for their experiences.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2  LITERATURE SURVEY</td>
<td>4</td>
</tr>
<tr>
<td>3  THE PROPOSED METHOD</td>
<td>7</td>
</tr>
<tr>
<td>4  CONCLUSIONS</td>
<td>26</td>
</tr>
<tr>
<td>APPENDIX A  PROGRAM</td>
<td>29</td>
</tr>
<tr>
<td>APPENDIX B  SOME RESULTS</td>
<td>51</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>67</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2 LITERATURE SURVEY</td>
<td>4</td>
</tr>
<tr>
<td>3 THE PROPOSED METHOD</td>
<td>7</td>
</tr>
<tr>
<td>3.1 Objective</td>
<td>7</td>
</tr>
<tr>
<td>3.2 Materials And Experimental Methods</td>
<td>7</td>
</tr>
<tr>
<td>3.3 Image-file Processing</td>
<td>7</td>
</tr>
<tr>
<td>3.4 Preprocessing</td>
<td>10</td>
</tr>
<tr>
<td>3.5 Feature Extraction</td>
<td>12</td>
</tr>
<tr>
<td>3.6 Extraction of Slant Strokes</td>
<td>21</td>
</tr>
<tr>
<td>3.7 Classification</td>
<td>23</td>
</tr>
<tr>
<td>4 CONCLUSIONS</td>
<td>26</td>
</tr>
<tr>
<td>4.1 Results</td>
<td>26</td>
</tr>
<tr>
<td>4.2 Discussion</td>
<td>26</td>
</tr>
<tr>
<td>4.3 Conclusions</td>
<td>28</td>
</tr>
<tr>
<td>4.4 Suggestion</td>
<td>28</td>
</tr>
<tr>
<td>APPENDIX A PROGRAM</td>
<td>29</td>
</tr>
<tr>
<td>APPENDIX B SOME RESULTS</td>
<td>51</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>67</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Original Image of Chinese Character</td>
<td>9</td>
</tr>
<tr>
<td>2 Preprocessing Result of Figure 1</td>
<td>11</td>
</tr>
<tr>
<td>3 Noisy Result of Vertical Feature Extraction</td>
<td>14</td>
</tr>
<tr>
<td>4 Noisy Result of Horizontal Feature Extraction</td>
<td>15</td>
</tr>
<tr>
<td>5 Satisfied Result of Vertical Feature</td>
<td>19</td>
</tr>
<tr>
<td>6 Satisfied Result of Horizontal Feature</td>
<td>20</td>
</tr>
<tr>
<td>7 Two Slant Strokes Extracted</td>
<td>22</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

Optical Characters Recognition enables a computer to recognize the characters written or printed on a paper or certain facility. It is a branch of Pattern Recognition, whose goal is to make a computer to distinguish among objects. The ambition of this domain is Computer Vision, which makes efforts on how a computer can "see" a 3D object in the real world as we, the humans, do. The basic common principles are how to define an object being recognized, to extract useful features from it, and to match the patterns.

Two of the important and practical applications of Optical Characters Recognition are for the post office, and for the blind. The others include automatic data entry, document processing, etc.

According to the input methods and facilities, the Optical Character Recognition, or OCR, can be classified into two classes, on-line, and off-line. Usually, the characters are input one by one in an on-line OCR, page by page in an off-line OCR.

The objects being recognized in an off-line OCR may be divided to four kinds: machine-printed, hand-printed, hand-written, and script characters. The hand-printed characters are written carefully. There are always much more different writing habits of certain writers existed in hand-written
Machine recognition of hand-printed or written Chinese characters is very difficult due to the following factors:

1. High complexity exists in the hand-printed or written characters because of the variation caused by the writing instruments, paper surfaces, social environment, and the writing habits, education background, mood, and health of the writers.

2. Compared to the total number (26x4) of English characters, the total number of Chinese characters is very huge. It is more than 40,000 totally, and 5,401 commonly used.

3. Almost 10% of Chinese characters such as 龍, 龜, and 墨 have more than 20 strokes.

4. Some characters are very similar, e.g. 己, 已, 巳, and 日, 日.

Generally, the methods used in an off-line OCR may have image file processing, binarization, text/graphics/image segmentation, characters segmentation, thinning, normalization, feature extraction, coarse classification, and fine classification.

The proposed methods are different from the above in some degree. Instead of the thinning and normalization, dilation are used. And this thesis focuses mainly on the feature extraction of Chinese characters. Besides, a set of simplified matching rules is created to verify the proposed
system that can work in an acceptable recognition rate and speed. However, for the purpose of commercialization, the matching algorithm should be revised and expanded.
CHAPTER 2

LITERATURE SURVEY

Geometrical and topological feature analysis method may represent characters' global and local properties including strokes, bays, end points, intersection of line segments, loops, and stroke relations, angular properties, sharp protrusions. These features have high tolerances to distortion, style variation, and some degree of rotation.

The methods of stroke extraction using traditional image processing have been studied. Chang [6] proposed an accumulated and learnable multifont (printed) OCR, extracting the strokes by choosing the maximum run length of each object pixel in 4 directions of degrees 0, 45, 90, and 135. But it is not good at distinguishing from the similar characters.

Another Chang [7] proposed a method of strokes extraction of Chinese characters by selecting the largest run length of every object pixel from 8 directions of degrees 0, 22.5, 45, 67.5, 90, 112.5, 135, and 157.5. Its disadvantage is too sensitive. And the time complexity is not good for a non parallel computer.

Chao [8] proposed a new stroke extraction method for hand-written characters based on the adjacency, branching, width, and length of runs of object pixels in horizontal direction only. The result of stroke extraction looks good, however, he did not prove whether or not the feature
extrated in his method can be applied on the matching process correctly and efficiently.

Chen[14] proposed a method of recognition of handwritten Chinese characters via short-line segments representing the strokes. The recognition rate is 90.88%, and the speed is 8.73 seconds per character.

The neural network applied on hand-printed or written Chinese characters recognition requires a large scale of neocognitron, and it is time-consuming to design the training patterns for feature extraction. Besides, the recognition rate is not ideal for the pattern which has not been trained. The parallel processing potential is an important characteristic of neural network, however it can not be developed easily on the popular personal computer at present. For illustration, it takes 28 seconds to recognize the hand-printed Chinese character with 86% recognition rate even on an IRIS workstation according to [11]. In [12], the recognition rate is 75% without rejection in the speed of 2 seconds per handwritten Chinese character on an IBM PC 386. In [13], 230 seconds are required even for the recognition of a handwritten capital English character on an IBM PC 386.

Context or language model [9][10] is also introduced into the OCCR. It is expected to help the recognition process to select the correct character from candidate sets produced by the coarse classification. However it can not ensure a satisfied recognition rate in a reasonable cost,
time and space complexity, because of the propagated error resulted from:

1. incorrect character in original text,
2. correct character in original text, but
   a. the correct character not included in the candidate sets, which is the result of the coarse classification,
   b. the inadequate decision policy of the language model itself,
   c. the incompleteness of the word base of the language model.
CHAPTER 3

THE PROPOSED METHOD

3.1 Objective

The objective of this thesis is to propose an improved traditional image processing method in which a popular personal computer can recognize off-line a scanned image of Chinese characters printed on a white paper by hand using a common blue ball-point pen.

3.2 Materials And Experimental Methods

Materials related to the experiment of the proposed method include an IBM PC 80486 DX33, a 300 dpi flat-bed grey level scanner, and a printer. The software is implemented in the C language.

The training and testing data are sampled from the same 500 Chinese characters written (hand-printed) by each of 15 persons with different ages and education background. The character size may be from 18x18 to 60x60 pixels. The size of 40x40 pixels is suggested.

The proposed system includes modules of image-file processing, preprocessing, feature extraction, post-processing, and classification.

3.3 Image-file Processing

In this thesis, the image of a Chinese character is scanned into a computer as the form of the grey level image file.
The coordinate system used is column major, and the origin is at the top-left corner.

The Constrained Run Length Algorithm [3][4][5] is used in the segmentation process.

Since so many tools can be found and used, this thesis doesn't focus on the simple process of open, display[2], close, and segmentation of a document image file.

```
  012345
Origin +---------> Y
    0
    1
    2
    3
    4
    5
    |
    |
    |
    X
```
Figure 1  Original image of Chinese character 木 represented by grey scale from 0, denoting black, to 255, denoting white.
3.4 Preprocessing

Before the feature extraction is performed, some preprocessing is required in order to delete noise and to improve the precision of feature extracted. First the thresholding and binarization process is applied.

By thresholding and binarization, the scanned-in grey level image file of a Chinese character might be transformed into a form of a binary image file, which would save more memory space. However, the image-file processing system is preferred to remain the form of grey level, because the level from 1 to 254 can be used to denote the intermediate result of some process on a pixel in this image besides of the level 0 denoting the object and 255 denoting the background.

Then the dilation is executed. Its goal is to make the object at least two pixels thick. The following is the algorithm of the dilation.

**Algorithm of Dilation:**

```algorithm
if ( (DilatedBinarizedOriginal[i-1][j]>1 )
    && (DilatedBinarizedOriginal[i][j]<1 )
    && (DilatedBinarizedOriginal[i+1][j]>1 ) )
   DilatedBinarizedOriginal[i-1][j] = 0;

if ( (DilatedBinarizedOriginal[i][j-1]>1 )
    && (DilatedBinarizedOriginal[i][j]<1 )
    && (DilatedBinarizedOriginal[i][j+1]>1 ) )
   DilatedBinarizedOriginal[i][j-1] = 0;
```
Figure 2 Preprocessing result of figure 1
3.5 Feature Extraction

There are many kinds of features which can be analyzed from Chinese characters. However, the main feature considered in this thesis is stroke including vertical, horizontal, and slant one.

Another feature considered is the number of the three kinds of strokes. The others are their relative position and length.

```
0 0 0 0 0 0-1 0 0 0 0
0 1 0 -1 1 0 0 1 0 0 1-1
0-1 0 0 0 0 0 0 0 0 0
```

Four masks as above are designed for the detection of upper, right-hand side, lower, and left-hand side boundary separately. By the following two algorithms, the vertical and horizontal strokes can be extracted.

Algorithm of vertical feature extraction:

```c
if ( (ResultOfMask1[i][j]=1) || (ResultOfMask3[i][j]=1) )
{
    VerticalFeature[i][j]=1; /* Hidden as black background */
}
else if (ResultOfPreprocessing[i][j]==255) /* white background */
{
    VerticalFeature[i][j]=0; /* black background */
}
else
```
Algorithm of horizontal feature extraction

if ((Result2[i][j]=1) || (Result4[i][j]=1))
{
    HorizontalFeature[i][j]=1; /* Hide as black background */
}
else if(Result0[i][j]>=100) /* white background */
{
    HorizontalFeature[i][j]=0; /* black background */
}
else
{
    HorizontalFeature[i][j]=255; /* white object */
}
Figure 3  Noisy result of vertical feature extraction
The feature extracted previously can not be satisfied. Some more post-processes including the deletion of single points, the breaking of the undesired connection, and the thresholding of the stroke length are required.

Algorithm of deletion of single points:

```c
if( (VerticalFeature[i-1][j-1] <= 1 )
&& (VerticalFeature[i ][j-1] <= 1 )
&& (VerticalFeature[i+1][j-1] <= 1 )
&& (VerticalFeature[i-1][j ] <= 1 )
&& (VerticalFeature[i ][j ] > 1 )
&& (VerticalFeature[i+1][j ] <= 1 )
&& (VerticalFeature[i-1][j+1] <= 1 )
&& (VerticalFeature[i ][j+1] <= 1 )
&& (VerticalFeature[i+1][j+1] <= 1 ))
    VerticalFeature[i][j] = 0;
if( (HorizontalFeature[i-1][j-1] <= 1 )
&& (HorizontalFeature[i ][j-1] <= 1 )
&& (HorizontalFeature[i+1][j-1] <= 1 )
&& (HorizontalFeature[i-1][j ] <= 1 )
&& (HorizontalFeature[i ][j ] > 1 )
&& (HorizontalFeature[i+1][j ] <= 1 )
&& (HorizontalFeature[i-1][j+1] <= 1 )
&& (HorizontalFeature[i ][j+1] <= 1 )
&& (HorizontalFeature[i+1][j+1] <= 1 ))
    HorizontalFeature[i][j] = 0;
```
Algorithm of breaking the undesired connection:

if( (VerticalFeature[i][j-3] > 1 )
    && (VerticalFeature[i][j-2] > 1 )
    && (VerticalFeature[i][j-1] > 1 )
    && (VerticalFeature[i][j] > 1 ))
    VerticalFeature[i][j] = 1;

if( (HorizontalFeature[i-3][j] > 1 )
    && (HorizontalFeature[i-2][j] > 1 )
    && (HorizontalFeature[i-1][j] > 1 )
    && (HorizontalFeature[i][j] > 1 ))
    HorizontalFeature[i][j] = 1;

if( (VerticalFeature[i-3][j-3] > 1 )
    && (VerticalFeature[i-2][j-2] > 1 )
    && (VerticalFeature[i-1][j-1] > 1 )
    && (VerticalFeature[i][j-1] > 1 ))
    VerticalFeature[i][j-1] = 1;

if( (HorizontalFeature[i-3][j-3] > 1 )
    && (HorizontalFeature[i-2][j-2] > 1 )
    && (HorizontalFeature[i-1][j-1] > 1 )
    && (HorizontalFeature[i][j-1] > 1 ))
    HorizontalFeature[i][j-1] = 1;

if( (VerticalFeature[i+3][j-3] > 1 )
    && (VerticalFeature[i+2][j-2] > 1 )
    && (VerticalFeature[i+1][j-1] > 1 )
    && (VerticalFeature[i][j-1] > 1 ))
    VerticalFeature[i][j-1] = 1;
if((HorizontalFeature[i+3][j-3] > 1 )
   &&(HorizontalFeature[i+2][j-2] > 1 )
   &&(HorizontalFeature[i+1][j-1] > 1 )
   &&(HorizontalFeature[i][j] > 1 ))
   HorizontalFeature[i][j] = 1;

Then, the connected components and the coordinates of
the top, bottom, left-most, and right-most ends of them are
found out. From these coordinates, the relative length and
position can be easily computed.

If the length is less than a threshold value, then this
connected component might be a noise and should be deleted.
According to the observation and thousands of experiments,
this value depends on each individual character and is very
critical to feature extraction, and even to the success of
classification recognition. However, it is assigned initially a
value of one tenth of the average of width and length of the
object being recognized. If the processes of extraction,
matching, or recognition are not satisfactory, this
threshold value could be adjusted.
Figure 5  Satisfied result of vertical feature after post-processing
Figure 6  Satisfied result of horizontal feature after post-processing
3.6 Extraction of Slant Strokes

Extraction of slant strokes can be obtained from the original object excluded the vertical and horizontal strokes.

Algorithm of extraction of slant strokes

```c
if ( VerticalFeature[i][j]>1) /* white object */
   ||(HorizontalFeature[i][j]>1)/* white object */
   ||(Result0[i][j]>1)) /* white background */
{
    OtherStroke[i][j]=0; /* black background */
}
else
{
    OtherStroke[i][j]=255; /* white object */
}
```

In addition to the post-processing of the deletion of single points, the breaking of the undesired connection, and the thresholding of stroke length, more procedures are required to separate the slant strokes if they are connected together, and find out the direction of them. These procedures are also based on comparing of the coordinates of the top, bottom, left-most, and right-most end points of slant strokes.
3.7 Classification

According to the number of vertical, horizontal, and slant strokes, the Chinese character can be classified coarsely. A simple algorithm of classification is used here to verify that the feature extracted previously are enough for an efficient Chinese character recognition system. However, for the practical application, this algorithm should be improved.

Algorithm of coarse classification:

switch (NumberOfVerticalStroke)
{
    ..
    case m:
        switch (NumberOfHorizontalStroke)
        {
            ..
            case n:
                switch (NumberOfSlant1Stroke)
                {
                    ..
                    case r:
                        Candidates of m vertical, n horizontal, and r slant stroke
                        Call fine-classification
                        break;
                }
            }
        }
    }
default:
printf(" Not Defined ");
break;
}
break;
..
default:
printf(" Not Defined ");
break;
}
break;
..
default:
printf(" Not Defined ");
break;
}

It is possible that several similar characters would be classified to the same class. In this case, a fine classification based on the relative position of stroke would be required.

For example, the characters 七, ㄆ, ㄆ, ㄆ, ㄆ, ㄆ, and 千 might be classified into the same class in the coarse classification due to having the same number of strokes, 1 vertical stroke, and 2 horizontal strokes. Fortunately, the relative position of these three strokes are different for each character. If not, they would even be impossibly distinguished by a Chinese.
Algorithms of fine classification are defined individually according to detail difference of the characters of the same coarse class.

Fine classification algorithm for 七, 士, 上, and 土:

```plaintext
if (LengthOfTopHorizontalLine < lengthOfBottomHorizontalLine)
    if (LeftEndOfTopHorizontalLine < LeftEndOfVerticalLine)
      return("土, Soil;");
    }
else
    {
      return("上, Up;" play);  
    }
else
    if (LeftEndOfBottomHorizontalLine < LeftEndOfVerticalLine)
      return("士, Shii;");
    }
else
    {
      return("七, 7;" play);  
    }
```
CHAPTER 4

CONCLUSIONS

4.1 Results

Excluding loading the original document image onto the screen and the segmentation, the computing time for feature extraction, classifications, and one file I/O depends on the image size and stroke number of each Chinese character, and it is about 0.3 seconds (30 seconds for 100 characters) on an IBM PC 80486 DX33 and C language environment. The recognition rate is 96% without rejection. The error is caused by high distortion. The training and testing data are carefully selected 100 similar and not so easy to distinguish characters out of from 500 characters written by each of 15 persons of different ages from 9 to 63 years old, and education background from elementary school to graduate school. The character size may be from 18x18 to 60x60 pixels. The size of 40x40 pixels is suggested.

Compared to [7][8][9][10][11][12][13][14], the result of the proposed methods is outstanding.

4.2 Discussion

Comparing with [6][7], more resonable range is assigned to each kind of strokes, -40 to 40 degrees for the horizontal strokes, 40 to 70 degrees for slant strokes, and 70 to 110 degrees for the vertical strokes.
The proposed method records the top, bottom, leftmost, and rightmost ends of each stroke extracted, and then exact position and relation can be computed. That is why the very similar characters can be distinguished. It is also the reason that the thinning and normalization are not adopted because the detail information might be neglected.

The advantages of the proposed method include:
1. size invariance,
2. tolerance to certain distortion of strokes,
3. ideal time complexity,
4. strong ability to distinguish very similar characters, and
5. 96% high precision. (Note: This precision could be higher as long as more fine classification is introduced for some certain characters.)
6. Further, the rule developed for coarse and fine classification based on stroke matching in this thesis is suitable for on line hand-printed Chinese character recognition, too.
7. The amazing performance of the proposed method is that it can endure the slope or rotation (in a certain degree) of the vertical and horizontal strokes.
8. The most important one is that it can detect the little difference among the very similar Chinese characters, e.g. 己, 巳, and 已.
9. Besides, the proposed method avoids the problems caused by thinning, normalization, and crossing.
However, the inflexibility for learning driven by the user is its disadvantage. This can also be overcome if an extra data base is built to store the learning result and act as the reserve for matching template in case that the fine classification is not defined perfectly by the manufacturer.

4.3 Conclusions

Feature extraction based on simple traditional image processing skill is more preferred than that based on neural network methods on the practical application of Optical Chinese Characters Recognition because it is expected that the parallel computers and languages are not as popular as personal computers, which is one kind of Von Neumann computers, in recent years.

Among so many Optical Chinese Characters Recognition based on traditional image processing skills, the proposed system has acceptable speed and ideal recognition rate.

4.4 Suggestion

The results for the sample data are acceptable, however, much more experiments for different characters are suggested since the total number of Chinese characters is so huge.

Besides, more efforts should be made on the storing, searching, and competing of the pattern base for matching to promote the performance of the whole recognition system.
APPENDIX

PROGRAM

/******* Version: 117 **********8,31***1994*************/

File : TIF117.c 8:46
Language: Turbo C
Machine: IBM PC AT 80486 DX 33
Programmer: Sunshine Chang

Master thesis of Mr. Sunshine Chang, 張善翔

...........................................

#include <io.h>
#include <mem.h>
#include <fcntl.h>
#include <string.h>
#include <graphics.h>
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include "c:\tc\tif\ocr.lib"

void main (int argc, char *argv[])
{
    printf("%d", argc);
    strcpy(fname, argv[1]);
    open_file();
    segmentation();
    feature();
}
matching();

/************ Version: 121  11,07***1994************
File : ocr.lib  12:17    included by tif117.c
Researched & developed by Mr. Sunshine Chang, 張善翔
************************** 山不在高 ************/
unsigned int tti, bbi;
unsigned int llj, rrj;
unsigned int ti, tj;
unsigned int bi, bj;
unsigned int li, lj;
unsigned int ri, rj;
unsigned char grey5[64][64];
unsigned int iiii=63, jjjj=63;
unsigned int xk=257, yk=1150;
unsigned int vk=12, hk=19, sk=14;
unsigned int vertion=121;
unsigned char grey0[64][64];
unsigned char grey1[64][64];
unsigned char grey2[64][64];
unsigned char grey4[64][64];
void Preprocess ()
{
    unsigned int i, j;
    float coefl[3*3] = { 0.0, 0.0, 0.0,
                        0.0, 1.0, 0.0,
                        0.0, -1.0, 0.0 };
float coef2[3*3] = { 0.0, 0.0, 0.0,
-1.0, 1.0, 0.0,
 0.0, 0.0, 0.0 };
float coef3[3*3] = { 0.0, -1.0, 0.0,
 0.0, 1.0, 0.0,
 0.0, 0.0, 0.0 };
float coef4[3*3] = { 0.0, 0.0, 0.0,
 0.0, 1.0, -1.0,
 0.0, 0.0, 0.0 };

for (i = 0; i <= iii; i++)
{
    for (j = 0; j <= jjj; j++)
    {
        if (*sp < threshold)
        {
            grey0[i][j] = 0;
        }
        else
        {
            grey0[i][j] = 255;
        }
        sp++;
    }
}
tag=iii; bbi=0; llj=jjj; rrj=0;
for (i = 0; i <= iii; i++)
{

for (j = 0; j <= jjjj; j++)
{
    if(grey0[i][j]<1)
    {
        if(i<tti) tti=i;
        if(i>bbi) bbi=i;
        if(j<llj) llj=j;
        if(j>rrj) rrj=j;
    }
}
}

for (i = tti+1; i <= bbi-1; i++)
{
    for (j = llj+1; j <= rrj-1; j++)
    {
        if(  (grey0[i-1][j]>1 )
            &&(grey0[i ][j]<1 )
            &&(grey0[i+1][j]>1 ))
            grey0[i-1][j]=0;
        if(  (grey0[i][j-1]>1 )
            &&(grey0[i][j ]<1 )
            &&(grey0[i][j+1]>1 ))
            grey0[i][j-1]=0;
    }
}

for (i = tti+1; i <= bbi-1; i++)
{

for (j = llj+1; j <= rrj-1; j++)
{
    grey1[i][j] = coef1[0] * grey0[i-1][j-1] 
    + coef1[1] * grey0[i-1][j ]
    + coef1[2] * grey0[i-1][j+1]
    + coef1[3] * grey0[i ][j-1]
    + coef1[4] * grey0[i ][j ]
    + coef1[5] * grey0[i ][j+1]
    + coef1[6] * grey0[i+1][j-1]
    + coef1[7] * grey0[i+1][j ]
    + coef1[8] * grey0[i+1][j+1];

grey2[i][j] = coef2[0] * grey0[i-1][j-1]
    + coef2[1] * grey0[i-1][j ]
    + coef2[2] * grey0[i-1][j+1]
    + coef2[3] * grey0[i ][j-1]
    + coef2[4] * grey0[i ][j ]
    + coef2[5] * grey0[i ][j+1]
    + coef2[6] * grey0[i+1][j-1]
    + coef2[7] * grey0[i+1][j ]
    + coef2[8] * grey0[i+1][j+1];

grey3[i][j] = coef3[0] * grey0[i-1][j-1]
    + coef3[1] * grey0[i-1][j ]
    + coef3[2] * grey0[i-1][j+1]
    + coef3[3] * grey0[i ][j-1]
    + coef3[4] * grey0[i ][j ]
    + coef3[5] * grey0[i ][j+1]
    + coef3[6] * grey0[i+1][j-1]
grey4[i][j] =
+ coef3[7] * grey0[i+1][j ]
+ coef3[8] * grey0[i+1][j+1];
coef4[0] * grey0[i-1][j-1]
+ coef4[1] * grey0[i-1][j ]
+ coef4[2] * grey0[i-1][j+1]
+ coef4[3] * grey0[i][j-1]
+ coef4[4] * grey0[i ][j ]
+ coef4[5] * grey0[i ][j+1]
+ coef4[6] * grey0[i+1][j-1]
+ coef4[7] * grey0[i+1][j ]
+ coef4[8] * grey0[i+1][j+1];

PreProcess2()
{
    unsigned int i, j;
    for (i = tti+1; i <= bbi-1; i++)
    {
        for (j = alalj+1; j <= rrj-1; j++)
        {
            if( (grey5[i-1][j-1] <=1 )
                &&(grey5[i ][j-1] <=1 )
                &&(grey5[i+1][j-1] <=1 )
                &&(grey5[i-1][j ] <=1 )
                &&(grey5[i ][j ] >1 )
                &&(grey5[i+1][j ] <=1 )

&& (grey5[i-1][j+1] <= 1 )
&& (grey5[i][j+1] <= 1 )
&& (grey5[i+1][j+1] <= 1 )

grey5[i][j] = 0;

if(  (grey6[i-1][j-1] <= 1 )
&& (grey6[i][j-1] <= 1 )
&& (grey6[i+1][j-1] <= 1 )
&& (grey6[i-1][j ] <= 1 )
&& (grey6[i ][j ] > 1 )
&& (grey6[i+1][j ] <= 1 )
&& (grey6[i-1][j+1] <= 1 )
&& (grey6[i ][j+1] <= 1 )
&& (grey6[i+1][j+1] <= 1 )
)

grey6[i][j] = 0;

for (i=tti; i<=bbi; i++)
{
for (j=llj+3; j<=rrj; j++)
{

if(  (grey5[i][j-3] > 1 )
&& (grey5[i][j-2] > 1 )
&& (grey5[i][j-1] > 1 )
&& (grey5[i][j ] > 1 )
)

grey5[i][j ] =1;

}
for (i=tti+3; i<=bbi; i++)
{
    for (j=llj; j<=rrj; j++)
    {
        if(  
            (grey6[i-3][j] >1 )
            &&(grey6[i-2][j] >1 )
            &&(grey6[i-1][j] >1 )
            &&(grey6[i ][j] >1 )
        )
            grey6[i ][j] =1;
    }
}
for (i=tti+3; i<=bbi; i++)
{
    for (j=llj+3; j<=rrj; j++)
    {
        if(  
            (grey5[i-3][j-3] >1 )
            &&(grey5[i-2][j-2] >1 )
            &&(grey5[i-1][j-1] >1 )
            &&(grey5[i ][j ] >1 )
        )
            grey5[i ][j ] =1;
        if(  
            (grey6[i-3][j-3] >1 )
            &&(grey6[i-2][j-2] >1 )
            &&(grey6[i-1][j-1] >1 )
            &&(grey6[i ][j ] >1 )
        )
            grey6[i ][j ] =1;
    }
}
for (i = tti; i <= bbi-3; i++)
{
    for (j = 1lj+3; j <= rrj; j++)
    {
        if( (grey5[i+3][j-3] >1 )
        &&(grey5[i+2][j-2] >1 )
        &&(grey5[i+1][j-1] >1 )
        &&(grey5[i ][j ] >1 ))
            grey5[i ][j ] =1;
        if( (grey6[i+3][j-3] >1 )
        &&(grey6[i+2][j-2] >1 )
        &&(grey6[i+1][j-1] >1 )
        &&(grey6[i ][j ] >1 ))
            grey6[i ][j ] =1;
    }
}
for (i = tti; i <= bbi-3; i++)
{
    for (j = 1lj+1; j <= rrj-6; j++)
    {
        if( (grey6[i ][j-1] >1 )
        &&(grey6[i ][j ] >1 )
        &&(grey6[i+1][j+1] >1 )
        &&(grey6[i+1][j+2] >1 )
        &&(grey6[i+2][j+3] >1 )
        &&(grey6[i+2][j+4] >1 )
        &&(grey6[i+3][j+5] >1 )}
void Extraction(unsigned int i, unsigned int j)
{
    if ((grey0[i][j]==1) || (grey3[i][j]==1))
    {
        grey5[i][j]=1;
    }
    else if(grey0[i][j]>0)
    {
        grey5[i][j]=0;
    }
    else
    {
        grey5[i][j]=255;
    }
    if ((grey2[i][j]==1) || (grey4[i][j]==1))
    {
        grey6[i][j]=1;
    }
    else if(grey0[i][j]>0)
void dN4SingleP()
{
    unsigned int i, j;
    for (i=tti+1; i<=bbi-1; i++)
    {
        for (j=1lj+1; j<=rrj-1; j++)
        {
            if(/* (grey3[i-1][j-1] <=1 ) */
                (grey3[i ][j-1] <=1 )
                /* &&(grey3[i+1][j-1] <=1 ) */
                &&(grey3[i-1][j ] <=1 )
                &&(grey3[i ][j ] >1 )
                &&(grey3[i+1][j ] <=1 )
                /* &&(grey3[i-1][j+1] <=1 ) */
                &&(grey3[i ][j+1] <=1 )
                /* &&(grey3[i+1][j+1] <=1 ) */
                grey3[i][j] = 1;
        }
    }
}
void feature ()
{
    PreProcess();
    j=0;
    fprintf(out," ");
    for (i=llj; i<=rrj; i++)
    {
        fprintf(out,"%d", j);
        j++;
        if(j>=10) j=0;
    }
    fprintf(out,"
");
    k=0;
    for (i=tti; i<=bbi; i++)
    {
        fprintf(out,"%d ", k);
        for (j = llj; j <= rrj; j++)
        {
            if(grey0[i][j]==255)
                fprintf(out,". ");
            else if(grey0[i][j]==1)
                fprintf(out,"_ ");
            else
                fprintf(out,"# ");
        Extraction(i, j);
    }
    fprintf(out," %d", k);
}
```c
k++;  
    if(k>=10) k=0;
}

fprintf(out,"\n\n ");  
j=0;
for (i=llj; i<=rrj; i++)
{
    fprintf(out,"%d", j);
    j++;
    if(j>=10) j=0;
}

fprintf(out,"\n\n\n%3d %3d %3d %3d
%3d %3d %3d %3d
%3d %3d %3d %3d
\n","tti-tti,rrj-llj,bbi-tti,rrj-llj,fname,xk,yk,gk,(bbi-tti+rrj-llj)/vk,(bbi-tti+rrj-llj)/hk,(bbi-tti+rrj-llj)/sk, vertion, bname);

fclose(out);
PreProcess2();
for (i = tti; i <= bbi; i++)
    for (j = llj; j <= rrj; j++)
        grey1[i][j] =0;
strcpy(bname, aname);
strcat(bname, ".8");
out8 = fopen(bname, "w");
fprintf(out8,"\n\n Class | Top | Bottom | Left-most|Right-most|
","tti-tti,rrj-llj,bbi-tti,rrj-llj,fname,xk,yk,gk,(bbi-tti+rrj-llj)/vk,(bbi-tti+rrj-llj)/hk,(bbi-tti+rrj-llj)/sk, vertion, bname);
```

The code snippet appears to be related to file operations and data processing, possibly for a graphics or image processing application. The comments and variable names suggest it deals with image data, possibly for a graphics or image processing application.
count = 0;
for (i=tti; i<=bbi; i++)
{
    for (j=llj; j<=rrj; j++)
    {
        if ( grey5[i][j] >= 255 )
        {
            ii = i; jj = j;
            ti = i; tj = j;
            bi = i; bj = j;
            lj = j; li = i;
            rj = j; ri = i;
            mark5(code" jj);
            if(((bi-ti)<=3) ||
                ((bi-ti)<=((bbi-tti)+(rr-llj))/vk))
                {
                    erase5(code, ii, jj);
                }
            else
            {
                if (code < 113)
                {
                    count = count + 1;
                    code = 254 - count;
                }
                code = code - 30;
            }
        }
    }
}
code = 254;
count = 0;
for (i=tti; i<=bbi; i++)
{
    for (j=llj; j<=rrj; j++)
    {
        if (grey6[i][j] >= 255)
        {
            ii = i; jj = j;
            ti = i; tj = j;
            bi = i; bj = j;
            lj = j; li = i;
            rj = j; ri = i;
            mark6(code, ii, jj);
            if (((rj-lj)<=3) || ((rj-lj)<=((bbi-tti)+(rrj-llj))/hk))
                erase6(code, ii, jj);
            else
                if (code < 113)
                {
                    count = count + 1;
                    code = 254 - count;
                }
        }
    }
}
code = code - 30;

strcpy(bname, aname);
strcat(bname, ".5");
out5 = fopen(bname, "w");
strcpy(bname, aname);
strcat(bname, ".6");
out6 = fopen(bname, "w");
j=0;
for (i=llj; i<=rrj; i++)
{
    fprintf(out5, "%d", j);
    fprintf(out6, "%d", j);
    j++;
    if(j>=10) j=0;
}
fprintf(out5, "\n");
fprintf(out6, "\n");
k=0;
for (i=tti; i<=bbi; i++)
{
    fprintf(out5, "\n%d ", k);
    fprintf(out6, "\n%d ", k);
    for (j=llj; j<=rrj; j++)


```c
{
    if (grey5[i][j]==1)
    {
        fprintf(out5,"-" );
    }
    else if (grey5[i][j]==0)
    {
        fprintf(out5,"." );
    }
    else
    {
        fprintf(out5,"V" );
    }
    if (grey6[i][j]==1)
    {
        fprintf(out6,"-" );
    }
    else if (grey6[i][j]==0)
    {
        fprintf(out6,"." );
    }
    else
    {
        fprintf(out6,"H" );
    }
    if ((grey5[i][j]>1)||(grey6[i][j]>1)||(grey0[i][j]>1))
```
{  
grey3[i][j]=0;
}
else  
{
  
grey3[i][j]=255;
}
}
fprintf(out5," %d", k);
fprintf(out6," %d", k);
k++;
if(k>=10) k=0;
}
fprintf(out5,"\n\n ");
fprintf(out6,"\n\n ");
j=0;
for (i=llj; i<=rrj; i++)
{
  
  fprintf(out5,"%d", j);
  fprintf(out6,"%d", j);
  j++;
  if(j>=10) j=0;
}
fprintf(out5,"\n\n\n%3d %3d %3d %3d
%3d %3d %3d
%3d %3d %3d
%3d %3d %3d
%3d %3d %3d
%3d %3d %3d
fclose(out5);
fprintf(out6, "\n\n%3d %3d %3d %3d
%3d %3d %3d
%3d %3d %3d\n%s %3d %3d %3d
%d %s", tti-tti, llj-llj, bbi-tti, rrj-llj, fname, xk, yk, gk, (bbi-tti+rrj-llj)/vk, (bbi-tti+rrj-llj)/hk, (bbi-tti+rrj-llj)/sk, vertion, bname);
fclose(out6);
dN4SingleP();
code = 254;
count = 0;
for (i=tti; i<=bbi; i++)
{
    for (j=llj; j<=rrj; j++)
    {
        if ( grey3[i][j] >= 255 )
        {
            ii = i;  jj = j;
            ti = i;  tj = j;
            bi = i;  bj = j;
            lj = j;  li = i;
            rj = j;  ri = i;
            mark7(code, ii, jj);
            if (((bi-ti)<5)||
                ((bi-ti)<=((bbi-tti)+(rrj-llj))/sk))
            {
                erase7(code, ii, jj);
            }
        }
    }
}
else
{
    if (code < 113)
    {
        count = count + 1;
        code = 254 - count;
    }
    code = code - 30;
}
}
match();
fclose(out8);
strcpy(bname, aname);
strcat(bname, ".7");
out7 = fopen(bname, "w");
j=0;
fprintf(out7," ");
for (i=llj; i<=rrj; i++)
{
    fprintf(out7,"%d", j);
    j++;
    if(j>=10) j=0;
}
fprintf(out7,"\n");
k=0;
for (i=tti; i<=bbi; i++)
{
    fprintf(out7, "\n%d ", k);
    for (j=llj; j<=rrj; j++)
    {
        if (grey3[i][j]==1)
        {
            fprintf(out7, "\n\n ");
        }
        else if(grey3[i][j]==0)
        {
            fprintf(out7, ".");
        }
        else
        {
            fprintf(out7, "X");
        }
    }
    fprintf(out7, "\n\n ");
    k++;
    if(k>=10) k=0;
}
fprintf(out7, "\n\n ");
j=0;
for (i=llj; i<=rrj; i++)
{

fprintf(out7, "%d", j);
    j++;
    if (j >= 10) j = 0;
}
fclose(out7);
APPENDIX B

SOME RESULTS

Example 1:

```
0123456789012345678901234567890123456789

0  ......................................  # # . ......................................  0
1  ......................................  # # .............................................  1
2  ......................................  # # .............................................  2
3  ......................................  # # .............................................  3
4  ......................................  # # .............................................  4
5  ......................................  # # .............................................  5
6  ......................................  # # .............................................  6
7  ......................................  # # .............................................  7
8  ......................................  # # .............................................  8
9  ......................................  # # .............................................  9
0  ......................................  # # .............................................  0
1  ......................................  # # .............................................  1
2  ......................................  # # .............................................  2
3  ......................................  # # .............................................  3
4  ......................................  # # .............................................  4
5  ......................................  # # .............................................  5
6  ......................................  # # .............................................  6
7  ......................................  # # .............................................  7
8  ......................................  # # .............................................  8
9  ......................................  # # .............................................  9
0  ......................................  # # .............................................  0
1  ......................................  # # .............................................  1
2  ......................................  # # .............................................  2
3  ......................................  # # .............................................  3
4  ......................................  # # .............................................  4
5  ......................................  # # .............................................  5
6  ......................................  # # .............................................  6
0123456789012345678901234567890123456789

CoarseClassificationResult ==> 七, 土, 工, 上, 土
FineClassificationResult ==> 七
```
Vertical feature, represented by symbol V, of example 1

```
0123456789012345678901234567890123456789
0 ............................VV.................. 0
1                   V............................... 1
2                   V............................... 2
3                   VVV........................... 3
4                   VVV........................... 4
5                  VVV........................... 5
6                  VVV........................... 6
7                  VVV........................... 7
8 _________  V______________________ 8
9 _________  V______________________ 9
0                VVV____________________ 0
1                VVV____________________ 1
2                VVV____________________ 2
3                VVV____________________ 3
4                VVV____________________ 4
5                VVV____________________ 5
6                VVV____________________ 6
7                VVV____________________ 7
8 _________  V______________________ 8
9 _________  V______________________ 9
0 _______________VV_________________ 0
1 _______________VV_________________ 1
2 _______________VV_________________ 2
3 _______________VV_________________ 3
4 _______________VV_________________ 4
5 _______________VV_________________ 5
6 _______________VV_________________ 6
0123456789012345678901234567890123456789
```
Horizontal feature, represented by symbol $H$, of example 1

0123456789012345678901234567890123456789
No slant feature of example 1

0123456789012345678901234567890123456789
Example 2:

```
0123456789012345678901234567890123456789
```

```
0
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
6
7
8
9
```

CoarseClassificationResult == > 七，士，工，上，士

FineClassificationResult == > 士
Vertical feature, represented by symbol V, of example 2
Horizontal feature, represented by symbol H, of example 2

```
0123456789012345678901234567890123456789
0
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
6
7
8
9
```

0123456789012345678901234567890123456789
No slant feature of example 2
Example 3:

```
012345678901234567890123456789012345678901234567
```

CoarseClassificationResult ==> 七，土，工，上，土

FineClassificationResult --> 土
Vertical feature, represented by symbol V, of example 3

012345678901234567890123456789012345678901234567

0          0
1          1
2          2
3          3
4          4
5          5
6          6
7          7
8          8
9          9
Horizontal feature, represented by symbol H, of example 3

01234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567
No slant feature of example 3
Example 4:

```
012345678901234567890123456789012345678901234567890
```

CoarseClassificationResult===> 七, 士, 工, 上, 土

FineClassificationResult===> 上
Vertical feature, represented by symbol V, of example 4
Horizontal feature, represented by symbol H, of example 4

012345678901234567890123456789012345678901234567890

```
0 .......................................................... 0
1 .......................................................... 1
2 .......................................................... 2
3 .......................................................... 3
4 .......................................................... 4
5 .......................................................... 5
6 .......................................................... 6
7 .......................................................... 7
8 .......................................................... 8
9 .......................................................... 9
0 .......................................................... 0
1 .......................................................... 1
2 .......................................................... 2
3 .......................................................... 3
4 .......................................................... 4
5 .......................................................... 5
6 .......................................................... 6
7 .......................................................... 7
8 .......................................................... 8
9 .......................................................... 9
0 .......................................................... 0
1 .......................................................... 1
2 .......................................................... 2
3 .......................................................... 3
4 .......................................................... 4
5 .......................................................... 5
6 .......................................................... 6
7 .......................................................... 7
8 .......................................................... 8
9 .......................................................... 9
0 .......................................................... 0
1 .......................................................... 1
2 .......................................................... 2
3 .......................................................... 3
4 .......................................................... 4
5 .......................................................... 5
```

012345678901234567890123456789012345678901234567890
No slant feature of example 4
REFERENCES


